

**Best  
Available  
Copy**

**AD-A280 434**

ADST/TR 94-003281



①

**ADVANCED DISTRIBUTED  
SIMULATION TECHNOLOGY  
ADVANCED ROTARY WING  
AIRCRAFT**

**SOFTWARE REUSABILITY  
REPORT**

Loral Systems Company  
12151-A Research Parkway  
Orlando, Florida 32826-3283

**DTIC**  
ELECTE  
JUN 14 1994  
**S F D**

April 8, 1994

Contract No. N61339-91-D-0001  
ARWA - Delivery Order No. 0048  
CDRL A002

DTIC QUALITY INSPECTED 2

Prepared for:

This document has been approved  
for public release and sale; its  
distribution is unlimited.

Simulation Training and Instrumentation Command  
Naval Air Warfare Center  
Training Systems Division  
12350 Research Parkway  
Orlando, FL 32826-3224

SSA 94-18198



94 6 13 000

REPORT DOCUMENTATION PAGE			Form approved OMB No. 0704-0188	
<small>Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget Project (0704-0188), Washington, DC 20503.</small>				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 08 April 1994		3. REPORT TYPE AND DATES COVERED
4. TITLE AND SUBTITLE ADST ARWA Software Reusability Report			5. FUNDING NUMBERS Contract No. N61339-91-D-0001 Delivery Order No. 0048	
6. AUTHOR(S) Karen Bourgeois Paul Kelly Robert Anschuetz Roger Branson				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Loral Systems Company ADST Program Office 12151-A Research Parkway Orlando, FL 32826			8. PERFORMING ORGANIZATION REPORT NUMBER ADST/TR-94-003281	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Simulation, Training and Instrumentation Command (STRICOM) c/o Naval Air Warfare Center, Training Systems Division 12350 Research Parkway Orlando, FL 32826-3224			10. SPONSORING ORGANIZATION REPORT NUMBER A002	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE  A	
13. ABSTRACT (Maximum 200 words) <p>The ADST ARWA Software Reusability Report provides a prediction of the effects on reusability of the software for the ARWA test bed and the ARWA Simulation System devices. The following three questions are addressed:</p> <ul style="list-style-type: none"> <li>- How reusable is the software when using the current development process?</li> <li>- What is the predicted effect on reusability of implementing the DA/CECOM/GA Tech recommendations?</li> <li>- What is the predicted effect on reusability of incorporating Ada style guidelines?</li> </ul> <p>Results of searches of source code and documentation libraries are included.</p>				
14. SUBJECT TERMS			15. NUMBER OF PAGES 55	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	17. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	17. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

# TABLE OF CONTENTS

1.0	Scope.....	1
1.1	Purpose.....	1
1.2	Background.....	1
1.3	Document overview.....	2
2.0	Referenced documents.....	2
3.0	Procedures.....	3
3.1	Description of current reuse efforts.....	3
3.2	Description of reuse options.....	11
3.2.1	Independent study suggestions.....	11
3.2.2	Ada style guidelines.....	13
3.2.3	Port to Ada.....	14
3.2.4	Domain analysis.....	14
3.2.5	Object-Oriented design conversion.....	14
3.3	Reuse level analysis.....	15
3.3.1	Assumptions.....	15
3.3.2	Reuse level model.....	16
3.3.3	Procedures.....	16
3.4	Reuse quality analysis.....	16
3.4.1	Assumptions.....	17
3.4.2	Reuse quality model.....	17
3.4.3	Procedures.....	18
3.5	Reuse cost impact analysis.....	18
3.5.1	Assumptions.....	18
3.5.2	Reuse cost model.....	19
3.5.3	Procedures.....	20
3.6	Reuse schedule impact analysis.....	21
3.6.1	Assumptions.....	21
3.6.2	Reuse schedule impact model.....	21
3.6.3	Procedures.....	21
4.0	Results.....	21
4.1	Reuse level analysis.....	21
4.2	Reuse quality analysis.....	22
4.3	Reuse cost impact analysis.....	23
4.4	Reuse schedule impact analysis.....	25
4.5	Summary.....	26
5.0	Conclusions and recommendations.....	26
5.1	Summary of conclusions and recommendations.....	26
5.2	Lessons Learned.....	27
6.0	Notes.....	27
6.1	Glossary.....	27
6.2	Acronym List.....	29

Accession For	
NTIS CRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification .....	
By <i>perlti</i>	
Distribution /	
Availability Codes	
Dist	Avail and/or Special
A-1	

## LIST OF FIGURES

Figure 1.	ARWA and ModSIM Architectures.....	5
Figure 2.	ADST Software Engineering Process Model.....	10
Figure 3.	Sample Kiviat Diagram.....	18
Figure 4.	Quality Results Kiviat Diagrams.....	24

## LIST OF TABLES

Table 1.	Reuse LOC Estimates for Common ARWA SS.....	6
Table 2.	Reuse LOC Estimates for RAH-66 Kit.....	8
Table 3.	Reuse LOC Estimates for AH-64D Kit.....	8
Table 3.1	Reuse LOC Estimates for Base and Kit.....	9
Table 4.	Segments With Existing Reusable Software.....	13
Table 5.	Example Loral Reuse Cost Schema.....	19
Table 6.	Reuse Cost and Productivity Scale .....	20
Table 7.	Reuse Quality Analysis Results .....	23
Table 8.	Reuse Cost Impact Analysis.....	25
Table 9.	Summary of Reuse Analyses.....	26

## APPENDIX A

10.	Introduction.....	A1
11.	Reuse Sources.....	A4
11.1	Asset Source for Software Engineering Technology (ASSET) .....	A4
11.1.1	Description.....	A4
11.1.2	Data Search .....	A4
11.1.3	Findings.....	A7
11.1.4	Rating .....	A7
11.2	Defense Software Repository System (DSRS).....	A7
11.2.1	Description.....	A7
11.2.2	Data Search .....	A7
11.2.3	Findings.....	A8
11.2.4	Rating .....	A8
11.3	Modeling and Simulation Information System (MSIS) .....	A8
11.3.1	Description.....	A8
11.3.2	Data Search .....	A8
11.3.3	Findings.....	A9
11.3.4	Rating .....	A9
11.4	Document Cataloging System (DOCATS).....	A9
11.4.1	Description.....	A9
11.4.2	Data Search .....	A9
11.4.3	Findings.....	A9
11.4.4	Rating .....	A9
11.5	Army Reuse Center (ARC).....	A10
11.5.1	Description.....	A10
11.5.2	Data Search .....	A10
11.5.3	Findings.....	A10
11.5.4	Rating .....	A10
11.6	Sherikon, Inc.....	A10
11.6.1	Description.....	A10
11.6.2	Data Search .....	A10
11.6.3	Findings.....	A10
11.6.4	Rating .....	A10
11.7	SPARTA, Inc. ....	A11
11.7.1	Description.....	A11
11.7.2	Data Search .....	A11
11.7.3	Findings.....	A11
11.7.4	Rating .....	A11
11.8	Public Ada Library (PAL) (Ada Software Repository).....	A12
11.8.1	Description.....	A12
11.8.2	Data Search .....	A12
11.8.3	Findings.....	A12
11.8.4	Rating .....	A12
11.9	Ada Joint Program Office (AJPO) and AdaIC.....	A12
11.9.1	Description.....	A12
11.9.2	Data Search .....	A12
11.9.3	Findings.....	A12
11.9.4	Rating .....	A12
11.10	National Technical Information Services (NTIS).....	A13
11.10.1	Description.....	A13
11.10.2	Data Search .....	A13
11.10.3	Findings.....	A13
11.10.4	Rating .....	A13
11.11	AdaNET.....	A13
11.11.1	Description.....	A13

11.11.2	Data Search .....	A14
11.11.3	Findings.....	A14
11.11.4	Rating .....	A14
12.	Conclusions .....	A14
13.	Bibliography.....	A14

## LIST OF TABLES

Table A1.	List of ARWA Models and Data .....	A1
-----------	------------------------------------	----

## APPENDIX B

20.	Introduction.....	B1
21.	Design Guidelines .....	B1
22.	Coding Guidelines.....	B2
22.1	General.....	B2
22.2	Ada Language .....	B3
23.	Bibliography.....	B4

## 1.0 Scope.

This software reusability study technical report is delivered under contract no. N61319-91-D-0001, Delivery Order Number 0048, for the Simulation, Training and Instrumentation Command (STRICOM), Naval Air Warfare Center, Training Systems Division, Orlando, FL. This study addresses the potential reusability of the Advanced Rotary Wing Aircraft (ARWA) test bed software in the Aviation Combined Arms Tactical Trainer (AVCATT) environment. Ground vehicles, other types of aircraft, or other types of military training simulators are not part of the scope of this study. However, it would be expected that any system with similar architecture could reuse significant portions of the software.

### 1.1. Purpose.

The purpose of this study is to provide the customer with enough information to make ARWA development decisions which may impact future development efforts in the ARWA/AVCATT realm. For instance, in future contracts STRICOM will be able to request a particular level or percentage of reuse when adding ARWAs to the Aviation Test Bed (AVTB) and be knowledgeable about the best approach toward reaching that level of reuse and productivity. To support this goal, this study contains quantitative data on the level of reusability potential of the ARWA system, including: costs, savings, and schedule impact.

In addition, the findings in this study shall also provide explicit information that will boost the level of software reusability in the ARWA system, especially during Phase II.

### 1.2. Background.

STRICOM requested that this study be performed during Phase I of the ARWA delivery order in parallel with the requirements and preliminary design phases of the effort. The Statement of Work (SOW) asked that this study "determine how reusable the ARWA software will be if developed in accordance with:"

- a. Existing ModSIM architecture
- b. Recommendations from the Institute for Defense Analyses (IDA), U.S. Army Communications-Electronics Command (CECOM), Georgia Institute of Technology (GIT), and Software Engineering Institute (SEI) studies (NOTE: The SEI study occurred after the SOW was written.)
- c. The Software Productivity Consortium's (SPC's) Ada Style Guide (SPC-91061-CMC).

NOTE: It is assumed in this report that "how reusable" refers to the quality (maturity level) and percentage of the product which will be reusable within the same domain if certain actions are taken, and that "ease of reuse" will translate into labor hours saved, i.e., productivity. Even though only software is mentioned, it is also assumed that the term "reusable" refers to any software workproduct, tool, or process that can be used again in another situation (i.e., software system, context, etc.) with 0 to 25 percent modification made to the existing object/idea that is going to be reused.

The second item to be addressed is the cost and schedule impact when implementing (a) and (b) above, plus the cost and schedule impact to reach higher levels of reuse via other reuse activities.

### 1.3. Document overview.

**Scope.** This section covers the purpose, scope, and background of this study.

**Referenced documents.** Several industrial publications and internal Loral documents are listed in this section as key references cited in the paper.

**Procedures.** The current and suggested reuse implementation plans are delineated in detail. The assumptions and procedures for analyzing the reuse level, resulting quality, cost, and schedule impact of each of these implementation activities are described.

**Results.** The results from analyzing each implementation activity is summarized in this section.

**Conclusions and recommendations.** This section summarizes the key conclusions and recommendations regarding the most cost effective approach for achieving the most reuse in the ARWA domain.

**Notes.** An acronym list and a short glossary of critical terms used in this paper are included in this section.

**Appendices.** These include detailed information on reuse guidelines, reuse tools, and models used for Verification & Validation (V&V) that will be used during subsequent phases of this project.

Appendix A describes the model/data searches for validating the two ARWA simulations, including the selection criteria.

Appendix B contains general reuse guidelines for design and coding in Ada. These are stored in the programmer's notebooks and have been shared with the designers.

## 2.0 Referenced documents.

The following documents are referenced within this report.

- |                 |  |
|-----------------|--|
| [Alexandris 86] | Alexandris, N. February 1986. "Adaptable Software and Hardware Problems and Solutions." <i>Computer</i> . Vol. 18. No. 2. pp. 29-39. |
| [Boeing 93]     | Boeing. 1993. "DARTS: A Domain Architecture for Reuse in Training Systems." Huntsville, Alabama.                                     |
| [CECOM 93]      | CECOM. April 1993. "White Paper for the Advanced Rotary Wing Aircraft Software Design Review." Leavenworth, Kansas.                  |
| [Gaffney 89]    | Gaffney, J., An Economics Foundation for Software Reuse, SW_REUSE_ECONM-89040-N, SPC, July 1989                                      |
| [GIT 93]        | GIT. April 1993. "Independent Assessment of the Advanced Rotary Wing Aircraft (RWA) Software Design for STRICOM." Atlanta, Georgia.  |

- [Hooten 89] Hooten, M. 1989. *Software Reuse Methodology and Checklists*. FACC-TR-1113. Ford Aerospace/Space Information Systems Division. (Now Loral Space Information Systems.) Houston, Texas.
- [IDA 93] Brykczynski, B. and D. Robert Worley. April 1993. "An Evaluation of the ModSIM Architecture and RWA Design." Institute for Defense Analyses. Alexandria, Virginia.
- [Lea 93] Lea, D. November 1993. *WISR'93 Design-for-Reuse Working Group Report*. Workshop on Institutionalizing Software Reuse held on November 1 - 4, 1993. IBM. Owego, New York.
- [Ogush 93] Ogush, M. 1993. "C Design and Coding Guidelines for Reuse." Hewlett-Packard. Palo Alto, California.
- [Parnas et. al. 89] Parnas, D., P. Clements, and D. Weiss. 1989. "Enhancing Reusability with Information Hiding." *Software Reusability Vol. I - Concepts and Models*. Association for Computing Machinery Press. pp. 141-157.
- [Parnas 72] Parnas, D. December 1972. "On the Criteria to be Used in Decomposing Systems into Modules." *Communications of the ACM*. Vol. 15.
- [SEI 93] SEI. July 1993. "Review of the Advanced Rotary Wing Aircraft Software Specification." Pittsburgh, Pennsylvania.
- [SPL 90] SPL. November 1990. *Corporate Productivity Lab Standards and Methods Document, Ada Standards, Volume 6*. SPL\_Ada\_STDS-90023-M. Version 1.0. Loral Software Productivity Laboratory. San Jose, California. section 1, p. 127-175. Section 2, pp. 45-55.

### 3.0 Procedures

This study was performed by software engineers with reuse expertise, coordinating inputs from cognizant Advanced Distributed Simulation Technology (ADST) software managers and designers, and from published data produced from in-house and industry reuse efforts. The first step was to establish a baseline of current reuse within the system being delivered by the Loral team and then define the additional reuse activity options that STRICOM should consider. These options stem from the SOW and current reuse philosophy. The next step was to evaluate the baseline and each option according to the resulting reuse maturity level, quality, cost, and schedule impact, if implemented. Numerical rating schemes were used to rate each option so that the best choice(s) would be easily identified by the highest total.

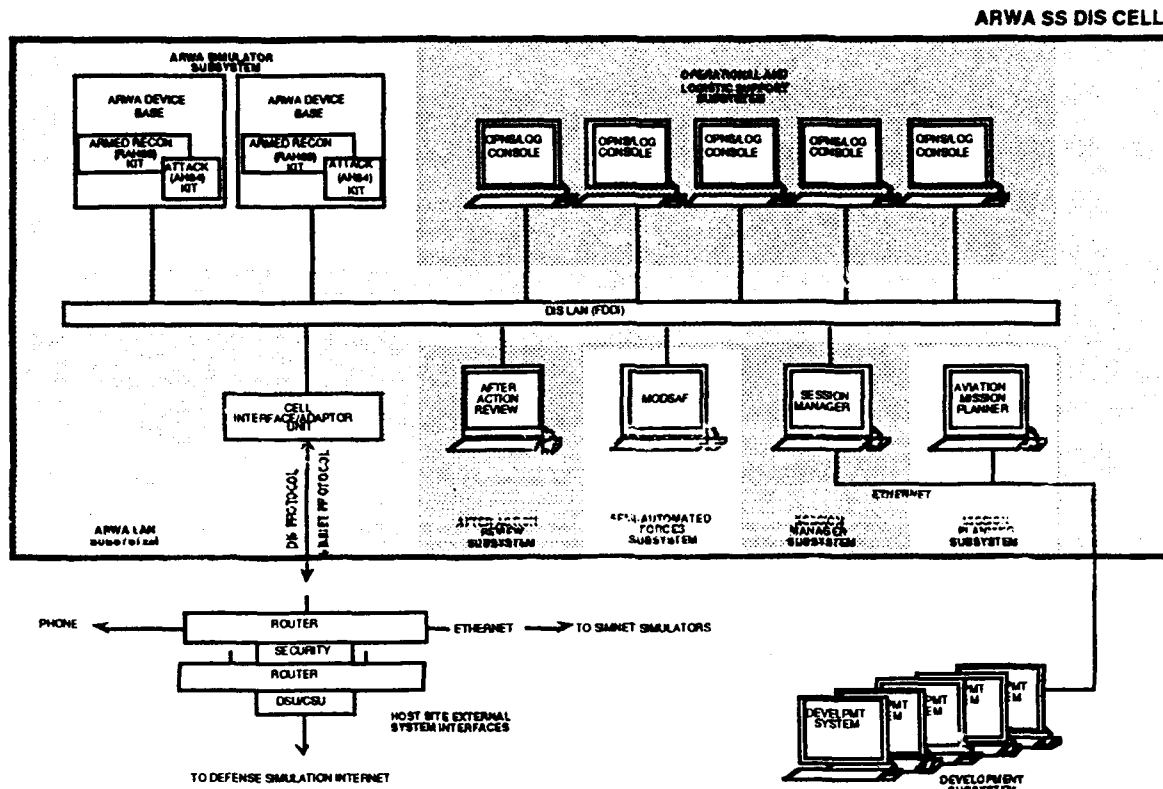
#### 3.1. Description of current reuse efforts.

The Loral team is applying six reuse techniques so that the delivered systems will contain fairly robust reuse features without affecting the current cost and schedule. These techniques are:

- a. Generic architecture. Conform to the generic Modular Simulation System (ModSIM) architecture as much as possible.
- b. External reuse. Search existing libraries for off-the-shelf software models/algorithms, specifications, test scenarios, database mapping data, etc. that could be reused in the ARWA Simulator System.
- c. Internal reuse. Reuse existing in-house designs and software from related simulation projects.
- d. Standard processes. Provide subcontractors with the same process and tool scripts used to count non-commented source lines of code (LOC).
- e. Uniform standards. Establish reuse design and coding principles to be used by the development team.
- f. Tool checker. Use software tools to check adherence to the Ada style guidelines.

Reasoning for generic architecture (a). ModSIM is a generic simulator architecture which defines a standard functional breakdown of a simulator system into 12 segments and defines standard interfaces between those segments. The 12 segments are as follows: Flight Station, Flight Controls, Flight Dynamics, Propulsion, Navigation/Communication, Weapons, Radar, Sensors, Physical Cues, Visual, Aircraft Survivability Equipment, Control, and Environment. One or more segments may be grouped on the same computational platform to form a module. Intersegment communication in ModSIM is accomplished by means of a message based architecture. Each segment communicates over a virtual network (VNET), which can be either through shared memory or over a physical network. By conforming to the ModSIM architecture, this simulator will be more easily maintainable in that those familiar with ModSIM's generic architecture will understand its design. The modular nature of the system facilitates accurate updates to the system, especially since the modules are highly cohesive and loosely coupled (i.e., have few intermodular interfaces).

Status of (a). According to the organizations that performed an independent evaluation of the ARWA architecture last year, the ARWA design conforms to the ModSIM architecture with some minor variations in the grouping of segments. The ARWA architecture separates the Visual and Flight Station segments into distinct modules - the Visual System Module (VSM) and the Flight Station Module (FSM), respectively - and groups the remaining ModSIM segments into the Simulator System Module (SSM). Figure 1 depicts both the ARWA architecture and the generic ModSIM architecture.



ARWA System Architecture

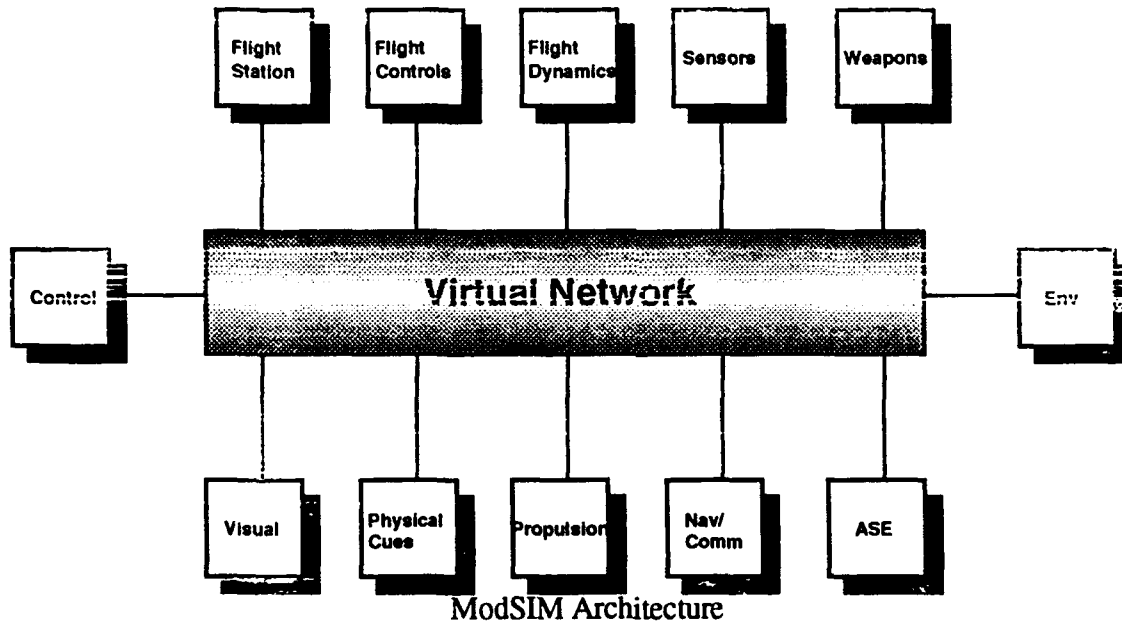


Figure 1. ARWA and ModSIM Architectures

Reasoning for external reuse (b). The assumption is that reused assets will be reusable in future applications of the simulator. This is true if the reused artifacts fulfill requirements that are not likely to change across ARWAs or over a long period of time within the aviation simulator training world.

Status of (b). The Loral team identified a list of software models which would be usable in the ARWA simulation system, as well as those needed to validate the accuracy of the simulation software. Refer to Appendix A for a listing of repositories searched and the results of those searches.

Reasoning for internal reuse (c). Internal reuse is defined as reusing software, data, and documentation from existing Loral, Boeing, and McDonnell Douglas Helicopter Systems (MDHS) efforts, as opposed to obtaining this information from external repositories. These systems were not necessarily designed for reuse, but reuse is relatively simple because the developers are already familiar with the architecture and software.

Status of (c). Boeing and MDHS have already identified much software which can be applied to the ARWA project, such as the Fly Real-Time (FLYRT) flight model and the Bus Interface Unit (VNET segment interface). Much of Boeing's reusable software has been obtained from Boeing Helicopter's Comanche Engineering Development Simulator. Much of MDHS's reusable software has been obtained from MDHS's Apache Engineering Simulator. Tables 1-3 identify the lines of code estimates as well as the amount of reusable software expected for the ARWA simulator system.

Reasoning for standard processes (d). In a multi-developer team environment, it is important that all parties follow the same processes in order to ensure that the delivered system's progress can be tracked and communicated in the same way.

Status of (d). One critical example, is the way LOC estimates were being made and reported. Loral provided a standard methodology for counting and reporting their progress using estimated and actual LOC data. One way to ensure accurate counts was to supply all of the subcontractors with the same in-house code counter scripts for Ada, FORTRAN, and C. This was very successful.

The Loral development process has also been communicated to the team via the process chart shown in figure 2. More is accomplished, more quickly when all of the team members use the same spiral development strategy.

Segment Name	Subsystem Name	Reused Code (LOC)	Total Code (LOC)	% Reused
VSM	VSM Network Interface	0	2,000	0 %
	VSM User Interface	0	6,000	0 %
	VSM Hardware Interface	0	10,000	0 %
	Process Scheduler	0	44,000	0 %
TOTAL		0	64,000	0 %

Table 1. Reuse LOC Estimates for Common ARWA SS

Segment Name	Subsystem Name	Reused Code (LOC)	Total Code (LOC)	% Reused
FSM	FSM Control	0	2,500	0 %
	Support Functions	0	4,500	0 %
	Aircraft Systems	0	250	0 %
	Real-Time	0	800	0 %
	I/O Linkage	0	920	0 %
	Control Load Linkage	0	460	0 %
	Flight Station Display Sys	0	1,500	0 %
<b>TOTAL</b>		<b>0</b>	<b>24,430</b>	<b>0 %</b>
SSM Control	Sim. Mod. & State	275	1,000	27 %
	Parameter Mod.	0	450	0 %
	Simulation Synchronization & Timing	0	660	0 %
	Executive	0	1,500	0 %
SSM TNE	Control	0	2,595	0 %
	Intervisibility	7,500	25,000	30 %
	Weapons	4,000	1,000	80 %
SSM BIU		4,840	4,840	100 %
<b>TOTAL</b>		<b>16,615</b>	<b>41,045</b>	<b>40 %</b>
Support Subsystems	Session Manager	0	5,438	0 %
	Operational & Logistic Support	3,168	5,280	60 %
	Mission Planning	2,900	5,800	50 %
	ModSAF	250,000	250,000	100 %
	After Action Review	7,000	10,000	70 %
	ARWA LAN	0	790	0 %
<b>TOTAL</b>		<b>263,068</b>	<b>281,179</b>	<b>94 %</b>
<b>GRAND TOTAL</b>		<b>279,683</b>	<b>410,654</b>	<b>68 %</b>

Table 1. Reuse LOC Estimates for Common ARWA SS [Continued]

Segment Name	Reused Code (LOC)	Total Code (LOC)	% Reused
Nav/Comm	0	2,100	0 %
ASE	0	2,125	0 %
Physical Cues	0	725	0 %
Sensors	0	2,815	0 %
Flight Controls	775	1,550	50 %
Weapons	500	2,000	25 %
Flight Dynamics	4,395	5,170	85 %
Propulsion	0	600	0 %
<b>TOTAL</b>	<b>5,670</b>	<b>17,085</b>	<b>33 %</b>

Table 2. Reuse LOC Estimates for RAH-66 Kit

Segment Name	Reused Code (LOC)	Total Code (LOC)	% Reused
Nav/Comm	1,820	2,800	65 %
ASE	320	640	50 %
Physical Cues	1,440	1,920	75 %
Sensors	1,120	2,240	50 %
Flight Controls	1,040	2,080	50 %
Weapons	1,000	4,000	25 %
Flight Dynamics	1,184	2,368	50 %
Propulsion	800	1,600	50 %
<b>TOTAL</b>	<b>8,724</b>	<b>17,648</b>	<b>49 %</b>

Table 3. Reuse LOC Estimates for AH-64D Kit

Module/Sub-System Name	Reused Code (LOC)		Total Code (LOC)		% Reused	
	Base	Kit	Base	Kit	Base	Kit
SSM Base Code	16,615		41,045		40 %	
SSM Kit Code <sup>1</sup>		7,200		17,365		41 %
FSM Base Code	0		20,760		0 %	
FSM Kit Code <sup>1</sup>		0		3,670		0%
VSM Base Code	0		60,800		0 %	
VSM Kit Code <sup>2</sup>		0		3,200		0 %
<b>Device Sub-Totals</b>	16,615	7,200	129,475	24,235	14 %	30 %
DIS Support Sub-Systems Base Code	263,068		281,179		94 %	
DIS Support Sub-Systems Kit Code		225		1,100		25 %
<b>Cell Sub-Totals</b>	279,683	7,425	410,654	25,335	68 %	29 %
<b>GRAND TOTAL</b>	<b>287,108</b>		<b>435,989</b>		<b>66 %</b>	

Notes: <sup>1</sup> Average of the AH-64D and RAH-66 software kits.

<sup>2</sup> Kit specific code estimated at 5 % of total VSM code.

**Table 3.1 Reuse LOC Estimates for Base and Kit**

Table 3.1 summarizes the reused line of code for the common software of the bases and the aircraft specific software code of the aircraft kits. The common software of the bases and some of the aircraft specific kit code is reusable for future experiments and aircraft implementations.

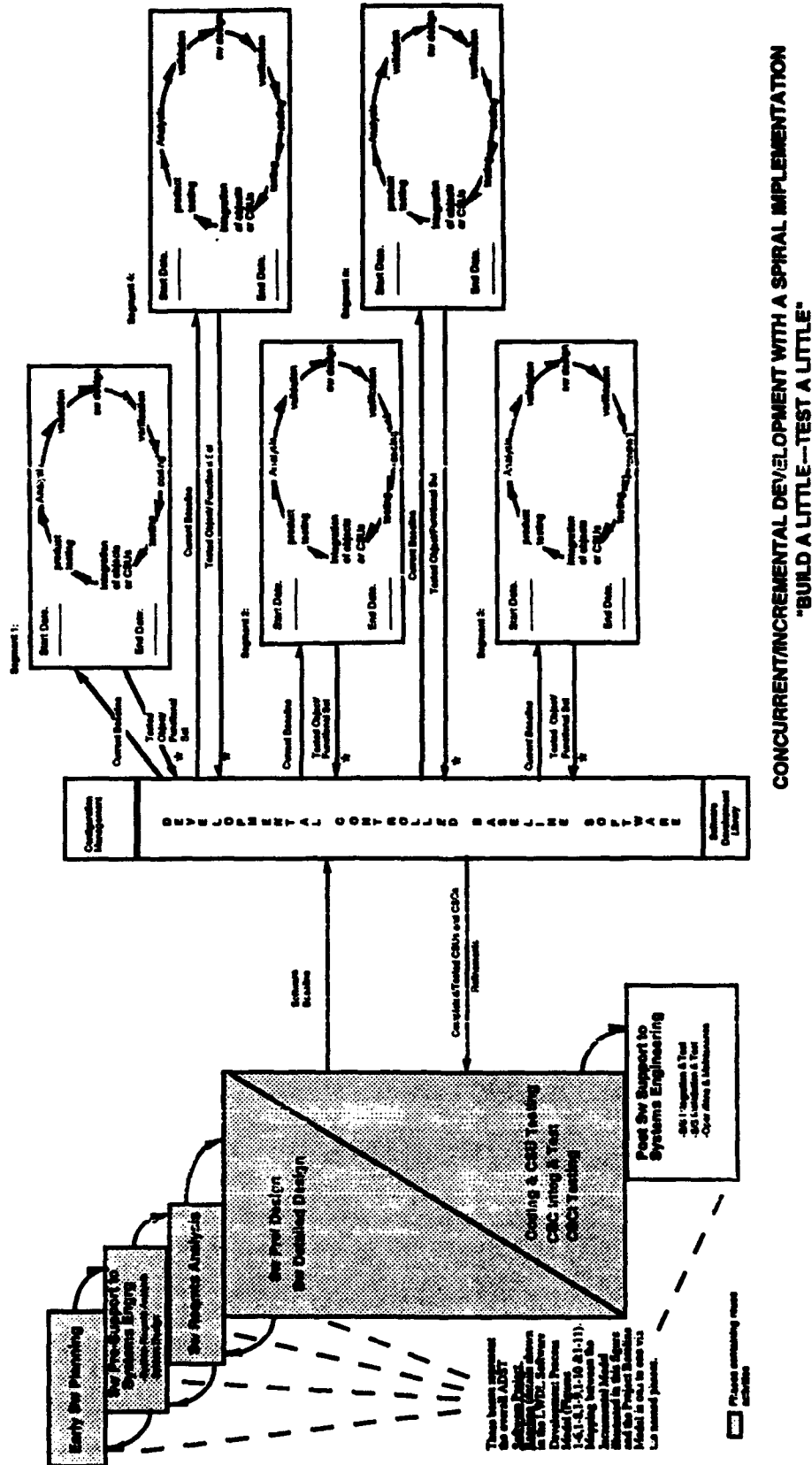


Figure 2. ADST Software Engineering Process Model

**Reasoning for uniform standards (e).** Standards ensure that a system will look and act in accordance with the requirements supported by the standard. In this case, a uniform set of reuse design and code standards will ensure that portions of the delivered system will be reusable and easily maintainable.

**Status of (e).** Each developer in the Loral team received a set of reuse design and code guidelines such as those contained in appendix B.

**Reasoning for tool checker (f).** A tool checker saves time in verifying code adherence to standards.

**Status of (f).** The Loral team plans to use the SPC's Ada Style Guidelines as contained in Loral's corporate Ada Style Guidelines [SPL 90]. Loral also has several software tools that automatically check the conformity of Ada source code to most of the Ada Style Guidelines.

### 3.2. Description of reuse options.

The models used in this study focus on the effects of reuse on productivity to produce the system. This study also extends the model to estimate future productivity resulting from specific reuse activities beyond the events mentioned in the SOW. Other factors considered in this study include the level of reuse maturity and the number of times something is reused.

Reuse implementation is more than just a technical issue, i.e., knowledge of the domain. Successful reuse entails proper management, guidelines, standard processes, training, tools, configuration management, and handling of legal issues.

#### 3.2.1. Independent study suggestions.

Four independent studies were funded by STRICOM and conducted by the following agencies: (1) Institute for Defense Analyses (IDA), Alexandria, VA [IDA 93], (2) Communications-Electronics Command (CECOM), Research, Development and Engineering Center, Software Engineering Directorate, Training & Maneuver Systems, Leavenworth, KS [CECOM 93], (3) Georgia Institute of Technology (GIT), Atlanta, GA [GIT 93], and (4) Software Engineering Institute (SEI), Carnegie Mellon University, Pittsburgh, PA [SEI 93]. These agencies addressed the same set of questions regarding the generic ModSIM System/Segment Specification (SSS) and ARWA designs. These questions dealt with the degree of design conformity to the ModSIM SSS, modularity, and adherence to object-oriented principles by the ARWA SS architecture in its incomplete state as of February 1993. Each independent evaluator was given a 22 volume set of documents which included Rotary Wing Aircraft (RWA) design data, unit development folders (UDFs), preliminary design review (PDR) slides, and preliminary design materials.

The first question asked "Is the System/Segment Specification for the Generic Modular Simulator - Specification #S495-10400C truly modular, reusable, and object-oriented in the design architecture presented?" The SEI report stated "The Generic Modular Simulator System (MSS), as presented in the System/Segmentation Specification is indeed modular," "The RWA Step 1 specification is a proof by existence that the specification for Generic MSS is reusable," and "The MSS specification is modular and has some attributes of the identity and classification object-oriented characteristics." The IDA report stated "We found the ModSIM architecture to be reasonably modular." The CECOM report stated "To the level of detail which was addressed in the System/Segment Specification, the design architecture is modular," "The design architecture outlined by the System/Segment Specification presents an architecture which could be easily tailored to particular flight simulator applications," and "The design architecture partitioned the system along

functional lines." The GIT report stated "The System/Segment Specification for the Generic Modular Simulator - Spec. # S495-10400C is truly Modular, Reusable and Object Oriented in the design architecture presented."

The second question asked "Does the design described in the RWA Step 1 report comply with the MODSIM guidelines/approach (defined in the System Segment Specifications for the Generic Modular Simulator - Specification #S495-10400C)?" The SEI report stated "The specification in the RWA Step 1 documentation complies, for the most part, with that presented in the MSS Generic specification." and "It is clear that RWA does adhere (both in spirit and in actuality, to the extent possible) to the MSS concepts/guidelines." The IDA report stated "The RWA design showed a high degree of compliance with the MODSIM architecture." The CECOM report stated "The Flight Station and Visual modules map closely to the modules defined in the MODSIM. However, the RWA SS creation of the Simulator System Module with 10 application segments deviates from the MODSIM guidelines." The GIT report stated "The design described in the RWA Step 1 report complies, in spirit, with the MODSIM guideline/approach."

The third question asked "Does the documentation provided, which represents the RWA design as accomplished by Loral/Boeing (i.e., unit development folders and other design documentation), comply with the MODSIM guidelines/approach? Is the RWA design modular, reusable and object-oriented?" The SEI report stated "The RWA specification closely follows the Generic MSS specification with respect to modularity, reusability, and use of an object-oriented approach, and the comments made about the MSS specification with respect to these properties also hold for the RWA specification." The IDA report stated "We found the RWA design, like the MODSIM architecture, to be reasonably modular but not based upon an object-oriented design." The CECOM report stated "In general, the UDFs were not at a point where an assessment of the code modularity could be performed.", "If indeed the code being imported is usable, the code should remain reusable for other applications. The design was not far enough into the details to determine if any new code generated would be reusable.", and "This approach does not map cleanly into object-oriented concepts." The GIT report stated "The documentation provided which represents the RWA design as accomplished by Loral/Boeing (i.e., Unit Development Folders and other design documentation) generally complies with the MODSIM guidelines/approach." and "The RWA design is Modular and Object oriented."

The fourth question asked "Does the System/Segment Specification for the Generic Modular Simulator - Specification #495-10400C, and consequently the RWA implementation of MODSIM, have adequate interface definitions to be implemented successfully in future simulator programs?" The SEI report stated "The requirements for future STRICOM simulator programs are not known by this review team, so the team cannot say specifically if the RWA approach will be adequate for these programs." The IDA report stated "... , we were unable to evaluate whether the MODSIM architecture provides adequate interface definitions to be implemented successfully in future simulator programs." The CECOM report stated "The interface definition is only at the top level. This does not provide the detailed information required to ensure successful implementation in either the RWA or future simulation models." The GIT report stated "The System/Segment Specification for the Generic Modular Simulator - Spec. #S495-10400C and consequently the RWA implementation of MODSIM provides a fairly detailed description of the system level requirements." and "... , the RWA design does indeed have a significant level of definition in its interface design."

Most of the agencies concluded that the ARWA SS architecture matched the generic ModSIM architecture, except that 10 independent segments have been grouped together as the SSM module. All of the agencies concluded that the architecture was indeed modular, but pointed out that the design did not fully adhere to some of the attributes of an object-oriented design.

Reusability of the ARWA was also addressed by the studies. The agencies generally came to the conclusion that if total software reuse is to be achieved by the ARWA project, STRICOM must include reuse requirements in the specification and Loral must provide more maintenance documentation to make reuse easier in the future.

Some segments have legacy code available from Loral, Boeing, and MDHS which can be reused in the ARWA program in their current state. The most reusable ARWA modules as determined by these studies and Loral's internal reuse estimates (tables 1 - 3) are listed in table 4 from the most reusable (listed first) to the least reusable (listed last). Those modules with close to no current reusable value in their current state are not shown.

ARWA Module Name	IDA	CECOM	GIT	Loral
Weapons	√	√	√	√
Flight Dynamics	√	√	√	√
Sensor Control	√	√	√	(√)
ASE	√	√	(√)	(√)
TNE (Environment)			(√)	√
Flight Control		(√)		√
BIU				√
Nav/Comm			(√)	
Visual Systems Module			(√)	
Propulsion			(√)	

NOTE: A parenthesized check mark (√) denotes that the module is only partially reusable.

**Table 4. Segments With Existing Reusable Software**

If, on the other hand, the Army is willing to switch to a more object-oriented architecture which is more conducive to reuse and maintenance, then the agencies suggested that the specification be modified to include reuse and object-oriented requirements. This would change the flavor of the contract from straight development into a reuse development project.

Since conversion to an object-oriented design is a separate option, the analyses will refer only to the structured design suggestions by these studies.

### 3.2.2. Ada style guidelines.

One of the options to consider is the use of the *SPC's Ada Quality and Style Guidelines for Professional Programmers*. Loral's software expertise provided some of the reusability and portability guidelines which were incorporated in the September 1991 version of this book. These guidelines have been in the *Corporate Standards and Methods Documents* since November 1990 and have been in use on Ada projects since then. Reusability is ensured by these guidelines because they address ways to handle Ada to promote

understanding and clarity, robustness, adaptability, independence, and key portability issues.

This option would ensure that Loral and subcontractors will use the guidelines during design and coding phases. The Grammatech Ada-Assured tool could be used to automatically check consistency with those guidelines, especially in the areas of portability and reusability.

### 3.2.3. Port to Ada.

Since Ada is the language of choice by the Government, and it has some desired features which support reuse, it would be valuable to port the entire simulator system to Ada. The Support Subsystems of the ARWA project, including ModSAF, includes 263,000 lines of reused 'C' code. The SSM common software contains 16,615 lines of reused FORTRAN code. The RAH-66 kit contains 25,770 lines of reused FORTRAN code. Systems requiring one type of compiler also reduce the cost of maintenance and improve system performance.

### 3.2.4. Domain analysis.

In order to achieve higher levels of productivity from reuse, one must work at higher levels of abstractions, such as the preliminary and detailed design levels.

Domain analysis of the ARWA is being performed to establish the common features among both the RAH-66 and AH-64D kits in order to determine both the static and variable aspects of each kit. From this information and the architectures of the ARWA kits for the RAH-66 and AH-64D, a generic architecture and data set for an ARWA is being created. The current designs, in some cases, use different models to accomplish the same result. Each commonality needs to be evaluated for genericity, testability, performance, and extensibility. The current schedule and funding permits this to some degree. The result is a design and data set template that contains the core set of features that are common to both the RAH-66 and AH-64D, with an optional set to accommodate unique features. Such templates would facilitate adding other rotary wing aircraft (such as the OH-58D) to the simulator structure.

Other domains such as U. S. Army training simulators that include ground-based vehicles and simulators could be explored in order to expand the distributed simulator network to interface with the ARWA SS and to accommodate combined arms military training.

### 3.2.5. Object-Oriented design conversion.

The structured ModSIM architecture constrains object-oriented design and reusability to a degree. Inheritance and information hiding are some of the features that would facilitate swapping of reusable building blocks that would fit into the architectural frameworks (system and database designs) of the simulator domain. These frameworks would easily be used in automatic simulator generators, like those on the market today, i.e., G2 made by Microsoft. The number of object-oriented methodologies with tool support is rapidly increasing. The technology is improving rapidly. Two acceptable methodologies are: Real-Time Object-Oriented Methodology (ROOM) and Schlaer-Mellor. Both have tool support, i.e., ObjectTime and Cadre, respectively.

One option would be to establish a parallel effort to convert the entire ARWA design and database into a totally object-oriented architecture. The alternative which is being pursued is to convert key portions of the design and database into an object-oriented structure. This is feasible if the portion was isolated enough from the rest of the design so as not to cause interface problems. For example, the VSM is essentially being defined from the ground

up, which affords an excellent opportunity to explore an object-oriented design. In this report, while discussing object-oriented conversion, the total conversion option is being addressed.

Another option would be to start implementing Ada 9X features in the design of the ARWA software in anticipation of its release. According to the Memorandum for Secretaries of the Military Departments Directors of the Defense Agencies concerning Early Use of Ada9X, dated March 9, 1994, the "revision of ANSI/MIL-STD-1815A (Ada83) has progressed to the point that it is nearly certain that the new version, referred to as Ada9X, will be approved by national and international standards bodies during 1994." The memorandum goes on to say that "early use of Ada9X provides access to the language's many enhancements, including full support for object-oriented programming, enhancements for real-time programming, and interfacing to other languages." Since validated versions of Ada9X will probably be available by the time the ARWA project is completed, conversion to a fully object-oriented design using Ada is a possibility for the ARWA program. In the meanwhile, steps can be made to design the ARWA software to increase the possibility of conformance to the Ada9X standard.

### 3.3. Reuse level analysis.

Each reuse option mentioned in sections 3.1 and 3.2 has been evaluated according to its reuse maturity level. Three levels are: opportunistic, systematic, and automatic generation.

The *opportunistic* level is the least mature level that yields the least amount of reusable products for the effort it involves. The user searches for reusable parts in an ad hoc manner, mainly at the lowest level of abstraction, i.e., code. There may or may not be a central reuse repository in which to find these parts. The ones that exist usually contain parts that are not relevant to the project, are not tailored for reusability, are not thoroughly tested, and do not follow the same standards. The user usually relies on past experience, private libraries, and notes to perform design and development activities.

The *systematic* level involves a well-defined and repeatable process with organizational commitments for funding, staffing, and incentives for production and use of reusable workproducts. Clear certification of parts and configuration management procedures are byproducts of systematic level reuse. In systematic reuse, the project schedules have more time allotted to the requirements and design activities, but shorter development times to accommodate more rapid prototyping at the framework level. Sophisticated library tools are not required, just logical directory structures with high quality parts relevant to the domain.

The *automatic generation* level cannot happen without the foundation of the systematic level. Systems are literally built while in the requirements and design phases with the aid of application generators. The most basic generators include 4GLs and User Interface Generators. The Cadre Teamwork CASE tool provides some basic code generation capabilities. More complex generation tools operate at higher-levels in order to hide the manual interconnection of components via a problem-oriented language, template, option filler, or visual programming environments (such as in the G2 tool, made by Microsoft). Internal domain expertise is needed to set up the application-specific parts and relate them to framework designs and specific requirements. The output is usually code and/or procedural calls in a higher order language.

#### 3.3.1. Assumptions.

The assumptions in this analysis are:

- a. There is a natural progression of mature reuse processes that involve increasing organizational commitment and more effective usage of the application experts' skills.
- b. The Government may become involved to incentivize such activities in order to make the producers more willing to produce reusable software.

### 3.3.2. Reuse level model.

The reuse level model focuses on the maturity of the reuse process. Three distinct levels (opportunistic, systematic, and automatic generation) are described in 3.3. If viewed on a continuum from least mature (opportunistic) to most mature (automatic generation), the following characteristics would apply:

#### Least Mature

No standards  
Manually search and use  
Small artifacts  
Low level of abstraction, e.g., code  
  
Nonrepeatable usage  
Different vocabulary  
No metrics  
Short-term reuse  
Unplanned  
Disjointed semi-reusable artifacts  
Low quality artifacts  
No training  
Domain knowledge not recorded  
Little management support  
No reuse organization  
Scattered focus on reuse applications  
Savings/costs not tracked  
Poor communication about reuse resources

#### Most Mature

Many standards  
Automatic assistance search  
Large artifacts  
High level of abstraction, e.g., frameworks  
Highly repeated usage  
Same vocabulary  
Reuse metrics  
Long-term reuse  
Planned  
Relational groupings artifacts  
High quality artifacts  
Training  
Domain knowledge recorded  
Management commitment  
Reuse organization  
Well-defined reuse areas  
Savings/costs tracked  
Good communication about reuse resources

With this continuum in mind, each reuse option was rated according to the following scale:

#### Reuse Level Rating Scale

- |                   |   |
|-------------------|---|
| 1 = Low           | (Opportunistic)                             |
| 2 = Average       | (Ad hoc with some systematic activities)    |
| 3 = Above Average | (Systematic)                                |
| 4 = Excellent     | (Systematic with some automatic generation) |
| 5 = Superior      | (Automatic generation)                      |

### 3.3.3. Procedures.

An expert assesses the reuse level of each option using the ranking values described above. The results are tabulated in a summary table.

### 3.4. Reuse quality analysis.

Each reuse option mentioned in sections 3.1 and 3.2 is evaluated according to the resulting quality of reuse.

### 3.4.1. Assumptions.

The assumptions during this analysis are:

a. Quality parameters that affect reusability are:

- Correctness
- Usability
- Adaptability
- Robustness
- Independence
- Understandability
- Portability
- Testability
- Accessibility
- Performance

*Correctness* is the degree a product fulfills its requirements in a consistent manner. This parameter is ensured by inspections, thorough testing, number of prior reuses, or some other certification process.

*Usability* is the extent to which the product will need to be modified to fit into another context. Minimal modification is desired.

*Adaptability* is the speed and ease in which a product may be tailored to fit into another context.

*Robustness* is the length of time a product is valuable as a reusable product, e.g., 5 years, 20 years, etc.

*Independence* is the degree to which the product is self-contained, i.e., is standalone and does not depend upon other artifacts for inputs.

*Understandability* is the level of clarity inherent in the product. The product is structured logically with complete documentation.

*Portability* is the ease in which a product is ported to another hardware platform or software language.

*Testability* is the ease in which a product is tested in a standalone or integrated situation.

*Accessibility* is the ease of acquiring the product for reuse. For example, a product has high accessibility when located within the project's local file system that is clearly labeled.

*Performance* is the amount of effect the product has on the system's performance when included in the system's framework.

### 3.4.2. Reuse quality model.

Since reuse quality has many facets, a Kiviat Diagram is used to visually show the differences in quality by showing whether a reuse activity would produce a certain reuse quality in the developed product. Each "spoke" in the diagram represents one of ten parameters and exhibits a quality rating. The quality rating scale for how well the reuse candidate fulfills the quality parameter is as follows:

Reuse Quality Rating Scale

- 1 = None
- 2 = Below average
- 3 = Satisfactory
- 4 = Above average
- 5 = Superior

The ratings are connected by a line and the resulting shape shows the quality profile. The larger the enclosed area, the higher the quality. An example of this diagram is shown in

figure 3. For ease of comparison with the other analyses, an average of the 10 parameters will be used to represent a particular reuse option.

### 3.4.3. Procedures.

An expert assesses the reuse quality of each option using the ranking values described above. The results are first tabulated in a Kiviati Diagram and then averaged for display in the summary table.

### 3.5. Reuse cost impact analysis.

Each reuse option mentioned in sections 3.1 and 3.2 is evaluated according to the resulting initial cost impact and future savings related to each option.

Reuse cost implementation depends upon the producer-user scenario. For example, there is a cost to making something more reusable and a cost for reusing something. The cost decreases when more mature levels of reuse are implemented and when artifacts are reused more than once.

#### 3.5.1. Assumptions.

The assumptions used in this analysis are:

- a. There are different productivity ratios for those who only produce reusable artifacts, produce and use once, use once, produce and use many times, and use many times.

## Sample Kiviati Diagram

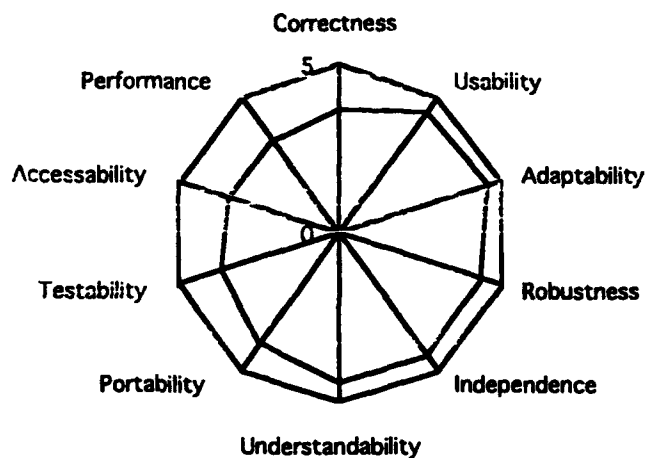


Figure 3. Sample Kiviati Diagram

- b. There is an initial cost impact to create a reusable artifact.
- c. The cost to use a reusable artifact may be greater or less than the cost to build it, depending upon the quality of the artifact and the skill of the user.
- d. Actual cost estimates cannot be calculated because the labor rates and processes are different for the producers and the consumers.

### 3.5.2. Reuse cost model.

The in-house cost model is similar to the commercial System Evaluation and Estimation of Resources (SEER) model. Part of the in-house costing process uses the SEER model results as a sanity check on the results. At the beginning of a project, the delivered product size is estimated and each LOC is associated with a productivity rate depending upon what type it is. The productivity ratio is defined as the ratio of hours per line of code. Higher productivity is associated with the lower values. Table 5 contains an example of this classification schema:

Code Type	Code Subtype	Productivity Ratio
New LOC	New application code	.95
	Non-delivered code	.35
Reused LOC	Added code	.32
	Changed code	.09
	Deleted code	.05
	Unmodified code	.03
	Ported code	.08
	COTS integration code	.25

**Table 5. Example Reuse Cost Schema**

To use this model would require data in smaller granularity than is available at this time. Therefore, for this study, a simpler approach is used that entails assessing the initial cost impact to implement a particular reuse option, estimating the productivity increase using the SPC's scale, and averaging the two rates.

Process or Tool	Productivity Increase	Magnitude of Cost
Compiler Library	1 (10 %)	2
Operating System	1 (10 %)	2
Scavenge	1 (10 %)	1
Junk Yard	1 (10 %)	1
Re-Engineering	3 (30 %)	1
Parts Library	3 (40 %)	2
Extensible Framework (based on Domain Analysis)	5 (120 %)	2
Synthesis (Automatic Generation + Domain Analysis)	5 (250 %)	3

**Table 6. Reuse Cost and Productivity Scale**

Table 6 shows the relative productivity increases and cost impacts for various types of reuse processes/tools based on data from the SPC [Durek 89] according to the following rating scales:

**Magnitude of Cost Impact Rating Scale:**

- 1 = Extremely High (more than 24 labor months)
- 2 = High (12 - 24 labor months)
- 3 = Medium (6 - 12 labor months)
- 4 = Low (1 - 6 labor months)
- 5 = Very Low (0 - 1 labor month)

**Productivity Increase Rating Scale:**

- 1 = Very Low (0 - 10 %)
- 2 = Low (11 - 30 %)
- 3 = Medium (31 - 50 %)
- 4 = High (51 - 100 %)
- 5 = Very High (greater than 100 %)

**3.5.3. Procedures.**

An expert assesses the reuse level of each option using the ranking values described above. The results are tabulated in a summary table.

### 3.6. Reuse schedule impact analysis.

This high level analysis indicates the impact on the project schedule for phase 2 of the ARWA project for each reuse option implementation mentioned in sections 3.1 and 3.2.

#### 3.6.1. Assumptions.

The assumptions in this analysis are that:

- a. Estimates are based on preliminary design information.
- b. Task dependencies that are logical.
- c. The schedule is impacted less if the activity is on a non-critical path.
- d. These are generic situations tacked on to current project schedules that may become a standalone project.

#### 3.6.2. Reuse schedule impact model.

In a more detailed analysis, each reuse option would have a skeleton work breakdown structure and a sample schedule would be plotted into PERT charts. The inputs would be validated by actual developers from the Loral team. For the sake of time, expert estimates are used to assess the amount of time and labor involved to implement each option and rank their impacts based on the following scale:

##### Schedule Impact Rating Scale

1 = Extremely high	(more than 1 year)
2 = Somewhat high	(6 months - 1 year)
3 = Medium	(3 - 5 months)
4 = Low	(2 weeks - 2 months)
5 = None	(0 - 2 weeks)

#### 3.6.3. Procedures.

An expert assesses the reuse level of each option using the ranking values described above. The results are tabulated in a summary table.

### 4.0 Results.

This sections contains the results of each of the four analyses used to evaluate the reuse implementation options described in section 3.2. The summary of the analysis results is contained in table 9. Equal weighting is assumed for each analysis. The option(s) with the highest average rating is the most optimal choice.

#### 4.1. Reuse level analysis.

The reuse level rating scale is as follows:

##### Reuse Level Rating Scale

1 = Low	(Opportunistic)
2 = Average	(Ad hoc with some systematic activities)
3 = Above Average	(Systematic)
4 = Excellent	(Systematic with some automatic generation)

5 = Superior

(Automatic generation)

Option 1 (current reuse activities) is ranked as a 2.5. This approach is more than just ad hoc because several systematic activities are happening such as, internal reuse from past projects, coordination of common software among the team, tracking the amount of reuse via LOC metrics, and following a generic simulator architecture. With the addition of some more reuse activities, this option would become a 3 (systematic level).

Option 2 (independent study suggestions) is ranked as a 2. The suggestions are a gentle push towards systematic reuse, but are not enough to achieve that level.

Option 3 (Ada style guidelines) is ranked as a 2. Incorporating standards for the code is just one activity out of many towards achieving systematic reuse.

Option 4 (port to Ada) is ranked as a 2. This small step has a positive impact on not only the current project, but future reuse opportunities in that the code will be incorporated more easily because it is in the same language. Performance will not degrade because of multiple compilers and so forth.

Option 5 (domain analysis) ranks as a 3. This is the core activity of systematic reuse. Domain expertise gets captured and efforts may be focused on products that bring the greatest return on investment.

Option 6 (object-oriented design conversion) is ranked as a 4. This conversion is a systematic activity that requires training and may involve software tools for quicker documentation. Object-oriented testing involves a different approach than testing structured code. More scenarios and a wider variety of tests are required.

#### 4.2. Reuse quality analysis.

The numerical results are contained in table 7. These results are graphically displayed via Kiviat Diagrams shown in figure 4. The quality gradually improves from option 1 to option 6. Options 3 and 4 are closely related. It is assumed that the port to Ada involves conformance to the Ada Style Guidelines. The quality rating is expressed in table 7 according to the following rating scale:

##### Reuse Quality Rating Scale

- 1 = None
- 2 = Below average
- 3 = Satisfactory
- 4 = Above average
- 5 = Superior

Parameters	Opt. 1: Current Actions	Opt. 2: Indep. Studies	Opt. 3: Ada Guide.	Opt. 4: Port to Ada	Opt. 5: Domain Anal.	Opt. 6: OOD
Correctness	2	4	3	3	5	5
Usability	4	4	4	5	5	5
Adaptability	4	4	5	5	5	5
Robustness	3.5	3	5	5	5	5
Independence	4	3	5	5	5	5
Understandability	2.5	4	5	5	5	5
Portability	3	3	5	5	3	5
Testability	3	4	3	4	4	4
	4	3	3	3	4	4
Performance	3	3	3	4	3	4
AVERAGE	3.3	3.5	4.1	4.4	4.4	4.7

Table 7. Reuse Quality Analysis Results

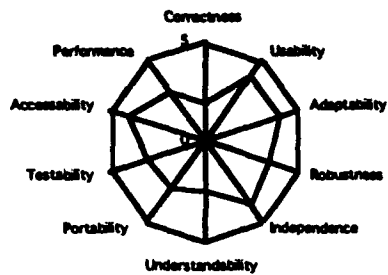
#### 4.3. Reuse cost impact analysis.

Cost impact was taken to be an average of the initial labor cost and the predicted reuse level. A summary of the reuse cost impact analysis is shown in Table 8 according to the following rating scales:

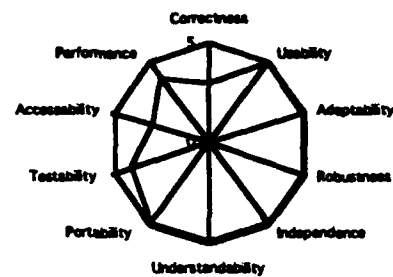
##### Initial Cost Impact Rating Scale:

- |                    |                             |
|--------------------|-----------------------------|
| 1 = Extremely High | (more than 24 labor months) |
| 2 = High           | (12 - 24 labor months)      |
| 3 = Medium         | (6 - 12 labor months)       |
| 4 = Low            | (1 - 6 labor months)        |
| 5 = Very Low       | (0 - 1 labor month)         |

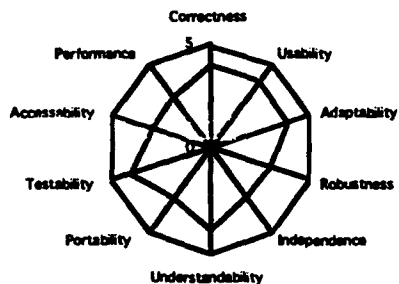
Option 1 - Current Activities



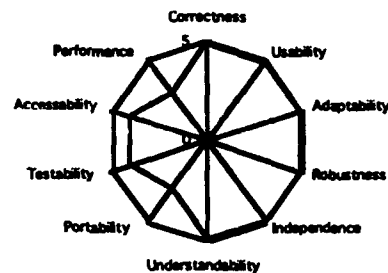
Option 4 - Port to Ada



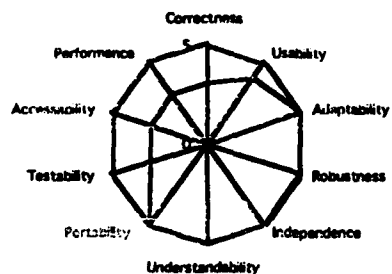
Option 2 - Independent Studies



Option 5 - Domain Analysis



Option 3 - Ada Style Guide



Option 6 - Object-Oriented

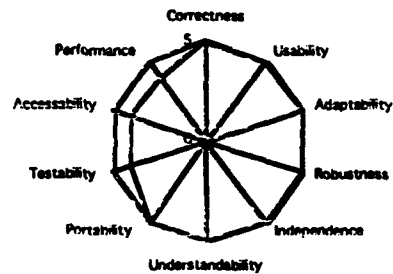


Figure 4. Quality Results Kiviat Diagrams

Predicted Reuse Level Rating Scale:

- |                   |   |
|-------------------|---|
| 1 = Low           | (Opportunistic)                             |
| 2 = Average       | (Ad hoc with some systematic activities)    |
| 3 = Above Average | (Systematic)                                |
| 4 = Excellent     | (Systematic with some automatic generation) |
| 5 = Superior      | (Automatic generation)                      |

Reuse Option	Initial Cost Impact	Predicted Reuse Level	Average Cost Input Rating
1. Current Reuse Activities	5	2.5	3.75
2. Independent Study Suggestions	5	2	3.5
3. Ada Style Guidelines	5	2	3.5
4. Port to Ada	4	2	3
5. Domain Analysis	3	3	3
6. Object-Oriented Design Conversion	2	4	3

Table 8. Reuse Cost Impact Analysis

#### 4.4. Reuse schedule impact analysis.

As a reminder, the rating scale for the reuse schedule impact analysis is as follows:

##### Schedule Impact Rating Scale

1 = Extremely high	(more than 1 year)
2 = Somewhat high	(6 months - 1 year)
3 = Medium	(3 - 5 months)
4 = Low	(2 weeks - 2 months)
5 = None	(0 - 2 weeks)

Option 1 (current reuse activities) is ranked as a 4. All of the activities are short tasks.

Option 2 (independent study suggestions) is ranked as a 4. The analysis of the functions requires the most amount of time.

Option 3 (Ada style guidelines) is ranked as a 4. Verification of following the guidelines takes the most time.

Option 4 (port to Ada) is ranked as a 1. Translation of more than 100K LOC requires a substantial effort.

Option 5 (domain analysis) ranked as a 3. Much of the functional analysis has already been done

Option 6 (object-oriented design conversion) is ranked as a 2. Much of the domain analysis and functional analyses can be used as a foundation and timesaver for this task. Also, in-house object-oriented experts may act as consultants to make this analysis go even more quickly. Designer/developer object-oriented training still needs to occur and this is what drives out the schedule. Training takes one week, but productivity would be initially slower until the concepts take hold; thus, the lower ranking.

#### 4.5 Summary.

A summary of all four analyses is shown in table 9.

Reuse Option	Reuse Level	Reuse Quality	Cost Impact	Schedule Impact	Average Rating
1. Current Reuse Activities	2.5	3.3	3.75	4	3.39
2. Independent Study Suggestions	2	3.5	3.5	4	3.25
3. Ada Style Guidelines	2	4.1	3.5	4	3.40
4. Port to Ada	2	4.4	3	1	2.60
5. Domain Analysis	3	4.4	3	3	3.35
6. Object-Oriented Design Conversion	4	4.7	3	2	3.43

Table 9. Summary of Reuse Analyses

#### 5.0 Conclusions and recommendations.

##### 5.1 Summary of conclusions and recommendations.

There are two views of reuse in this study: 1) using reusable artifacts to build and test the system being developed and 2) ensuring that a portion of the system will be reusable in the future. The Loral team must act both as a consumer and a producer in the reuse world. The reuse analyses performed in this study provide some guidance for accomplishing the most reuse (short-term and long-term) with the least impact to cost and schedule.

According to the results, the least productive option is to port the system to Ada and the next to least effective option is to follow the suggestions put forth in the independent study papers. The top options, beginning with the best choice, are: object-oriented conversion, adopting the Ada Style Guidelines, performing current reuse activities, and performing a domain analysis on the system. The top options improve the chances for long-term reuse. These are intermediate level (systematic) activities. If domain analysis and object-oriented design are performed, it will be feasible and cost-effective to perform automatic generation of training simulators using Commercial-Off-The-Shelf (COTS) tools and domain experts.

The most immediate and feasible activity for performing a domain analysis would be to start with the ARWA kits for RAH-66 and AH-64D to determine a generic architecture, modeling equations, and data format. Variable features would be noted for the object-oriented design.

The most logical means of transitioning to an object-oriented design is to pursue the Boeing Domain Architecture for Reuse in Training Systems (DARTS) methodology [Boeing 93]. The DARTS architecture is essentially a merging of Boeing's ModSIM architecture with the

Software Engineering Institute's Air Vehicle Structural Model (AVSM) architecture. The ModSIM architecture defines modular segments of a training system and the interfaces between those segments. The AVSM architecture defines the subsystems within each segment and the interfaces between those subsystems.

Thus far, the ARWA preliminary design effort has produced a ModSIM design, with clearly defined segments and interfaces. A transition from the ModSIM architecture to the DARTS architecture at this point would be relatively smooth since much of the internal workings of the segments have yet to be defined. Effort spent thus far on defining the system under the ModSIM architecture could be utilized completely in a transition to the DARTS architecture.

Transitioning to the DARTS architecture at this point in time makes a great deal of sense for the ARWA project. In order to keep the intrasegment functionality and interfaces consistent between MDHS and Boeing, a standard methodology needs to be chosen. Since DARTS is compatible with ModSIM, it meets the goal of providing standard intrasegment definitions while retaining the design worked on thus far. The DARTS architecture produces an object-abstracted design, and DARTS architected software will allow for reusable software within segments.

The modules of the ARWA SS which utilize existing reusable software in its current state include the Weapons, Flight Dynamics, Sensor Control and ASE modules. The TNE (environment) and Flight Control modules also have reuse potential. In order to encourage future reuse of these modules, the customer should require a separate Contract Data Requirements List (CDRL) item for these modules that contain reuse instructions, e.g., what to change or not change. The VSM and FSM have been designed to be reusable within the constraints of a ModSIM architecture, though no existing code has been identified to be reused. Reuse will come about through careful design and documentation.

## 5.2 Lessons Learned.

Appendix A describes the search for models and data to help validate the system. These resources uncovered some reusable modules for the ARWA project, but not as many as had been hoped for.

The best source for reusable artifacts were found within the contractor's software and documentation from previous related projects. Since there is a lot of internal reuse occurring, the customer should require a CDRL item that captures the reuse successes and failures.

The current ARWA approach is certainly a viable solution to producing reusable software at a reasonable cost. Changes to an object-oriented design through domain analysis have been shown to be cost efficient. The simplest and most effect means to transition to an object-oriented approach would be to incorporate the DARTS methodology.

## 6.0 Notes.

This section contains a glossary of key terms and an acronym list.

### 6.1 Glossary

**Automated Reuse.** Software reuse accomplished via the use of application generators to build new applications from high level descriptions. Examples include 4GLs and User Interface Generators.

**Domain Analysis.** The process of identifying, collecting, organizing, analyzing, and representing a domain model and software architecture from the study of existing systems underlying theory, emerging technology, and development theories within the domain of interest.

**External Reuse.** Reuse of workproducts produced in one project, consumed by another. External reuse level is measured by comparing units written against units taken from an explicit external library at that abstraction level.

**Framework.** A set of workproducts or infrastructure that behaves as a skeletal system or application and implements the common functionality in an architecture. A framework provides a shell for the systematic development and interconnection of workproducts, ensuring common appearance and behavior via use of common services.

**Generator.** A higher-level automatic builder that hides the manual interconnection of components using a problem-oriented language, template or option filler, or a visual programming environments. The generator enables concise specification of the desired (piece of the) application, and then generates appropriate code and/or procedure calls in some other language.

**Generic Application.** A customizable/extendible application that captures most of the interesting, common parts of an application domain. a complete application is built by adding missing parts, adjusting parameters, or selecting alternative components. It is often built upon an application framework. It can also be a prototypical or skeletal application, consisting of the infrastructure, some components, and some preset interconnection language scripts, to simplify the task of creating complete, conforming applications for some domain. This may be just a shell, into which additional components should be plugged to produce an executable application, or may be a trivial, but complete application that needs to be evolved into the final/desired/customized application via the addition or replacements of components and changes in interconnection language.

**Internal Reuse.** Avoiding redundant implementation of functionality within a single project by careful design and inspection at early stages such that selected components are identified for distinct uses within the project system or subsystem.

**Opportunistic Reuse.** Reuse through identification of previously unplanned-for opportunities to reuse workproducts.

**Reusability.** An attribute of software workproducts that measures the degree to which they can be used in more than one computer program or software system.

**Systematic Reuse.** The planned reuse of workproducts with a well-defined process and lifecycles, with commitments for funding, staffing, and incentives for production and use of reusable workproducts.

**6.2 Acronym List**

<b>ADST</b>	<b>Advanced Distributed Simulator Training</b>
<b>ARWA</b>	<b>Advanced Rotary Wing Aircraft</b>
<b>AVCATT</b>	<b>Aviation Combined Arms Tactical Trainer</b>
<b>CECOM</b>	<b>U.S. Army Communications-Electronics Command</b>
<b>COTS</b>	<b>Commercial-off-the-shelf</b>
<b>CSCI</b>	<b>Computer Software Component Item</b>
<b>FSM</b>	<b>Flight Station Module</b>
<b>GIT</b>	<b>The Georgia Institute of Technology</b>
<b>IDA</b>	<b>Institute for Defense Analyses</b>
<b>LOC</b>	<b>Lines of Code</b>
<b>ModSAF</b>	<b>Modular Semi-Automated Forces</b>
<b>ModSIM</b>	<b>Modular Simulator System</b>
<b>PERT</b>	<b>Program Evaluation and Review Technique</b>
<b>SEER</b>	<b>System Evaluation and Estimation of Resources</b>
<b>SEI</b>	<b>Software Engineering Institute</b>
<b>SPC</b>	<b>Software Productivity Consortium</b>
<b>SS</b>	<b>Simulator System</b>
<b>SSM</b>	<b>Simulation Software Module</b>
<b>SSS</b>	<b>System/Segment Specification</b>
<b>STRICOM</b>	<b>Simulation, Training, and Instrumentation Command</b>
<b>SW</b>	<b>Software</b>
<b>TNE</b>	<b>Tactical &amp; Natural Environment Module</b>
<b>TWSTIAC</b>	<b>Tactical Warfare and Simulation Technology Information Analysis Center</b>
<b>VSM</b>	<b>Visual System Module</b>
<b>WDL</b>	<b>Loral Western Development Labs</b>

## APPENDIX A

### REPOSITORIES CHECKED FOR REUSE INFORMATION

#### 10. Introduction

Loral has identified models for potential reuse in the ARWA simulation. From this model list, repository sites have been searched for availability to make an initial top-level judgment of reusability. Table 1 contains the list of models and data.

Numerous repositories for reusable data, documentation, and source code for the ARWA program have been searched. These include:

- 1) ASSET Source for Software Engineering Technology
- 2) Defense Software Repository System (DSRS)
- 3) Modeling and Simulation Information System (MSIS)
- 4) Document Cataloging System (DOCATS)
- 5) Army Reuse Center (ARC)
- 6) Sherikon, Inc.
- 7) Sparta, Inc.
- 8) Public Ada Library (PAL) (Ada Software Repository)
- 9) Ada Joint Program Office (AJPO) and AdaIC
- 10) National Technical Information Services (NTIS)
- 11) AdaNET

Various categories of information for each source are given as follows:

- |                |                                      |
|----------------|--------------------------------------|
| 1) Description | - A brief description of the source  |
| 2) Data Search | - Information about search performed |
| 3) Findings    | - Results of search                  |
| 4) Rating      | - Reusability rating of repository   |

The general criteria categories for rating the reuse data repositories were:

- 1) Available relevant software models
- 2) Available relevant documentation
- 3) Cost, data rights, and electronic access

A score of 0 (worst) to 10 (best) has been given to each category and a final score has been tabulated for each source.

Segment	System	RAH-66	AH-64D
Flight Control	Primary Controls	✓	✓
	Flight Director	✓	✓
	Landing Gear Doors	✓	
	Landing Gear	✓	✓
	Flight Controls Loading	✓	✓
	AFCs	✓	✓
	Velocity Stabilization	✓	

Table A1. List of ARWA Models and Data

Segment	System	RAH-66	AH-64D
Nav/Comm	HARS/AHRS	✓	INU
	DNS	✓	Integrated with GPS
	GPS	✓	✓
	ICS	✓	CCP
	VHF COMMS	ARC-186 ARC-201 HF	ARC-186 ARC-201
	UHF COMMS	✓ (2)	ARC-164
	Air Data	✓	ADS
	ATHS	EATHS up to 16K Baud	EATHS up to 16K Baud IDM
	Line-of-sight and range attenuation models and data	✓	✓
	Moving Map	✓	NAV/TSD
Weapons	Area Weapon System	20 mm gun	M-230E1 30 mm gun
	Aerial Rocket System	Hydra 70 2.75" RKTS MK-66 MPSM	2.75" RKTS MK-66 MPSM
	Point Target System	AGM-114 Hellfire Laser Seeker	AGM-114A Hellfire RF Seeker Laser Seeker
	Heat Seeking Missiles	ATAS	ATAS
	Hit/Kill Probability, models & data	✓	✓
Sensor	PNVS	NVPS	AN/AAQ-11 FLIR
	TADS	EOTADS AN/ASQ-170 FLIR DTV DVO LRF/D LST/IAT	AN/ASQ-170 FLIR DTV DVO LRF/D LST/IAT
	HIADSS		HIDU SSU DAP SEU DEU
	HIDSS	✓	
	MMW Radar	✓	FCR
	Sensor degradation based on atmospheric conditions including smoke, fog, and rain	✓	✓
	RFI	✓	✓
	Material Emissivity Model	✓	✓

Table A1. List of ARWA Models and Data [Continued]

Segment	System	RAH-66	AH-64D
Aircraft Survivability Equipment	Radar Warning	APR-39 APR-48	APR-39 (V) 1 (V) 2 APR-48
	Laser Warning	AVR-2	AVR-2
	Radar Warning	✓	
	Chemical Warning	✓	
	Radar Jammer	✓	ALQ-136 (V) 1/5
	IR Jammer	✓	ALQ-144 (V) 1/3
	Chaff	✓	M-130
	Flare	✓	M-130
Flight Dynamics	Equations of Motion	✓	✓
	Mass Properties	✓	✓
	Main Rotor Aerodynamics Blade element Rotor mapped disc	✓	✓
	Tail Rotor Aerodynamics	✓	✓
	Airframe Aerodynamics	✓	✓
	Ground Handling	✓	✓
Propulsion	Main and Tail Rotor Speeds	✓	✓
	Transmission	✓	✓
	Transmission Oil Temperature	✓	✓
	Transmission Oil Pressure	✓	✓
	Gas Generator/Power Turbine	T800	701C
	Engine Oil Temperature	✓	✓
	Engine Oil Pressure	✓	✓
	Engine Available Torque	✓	✓
	Fuel Usage	✓	✓
Physical Cues	Turbine Gas Temperature	✓	✓
	Environmental sounds and vibrations, ASE, Aircraft, Weapons, Explosions	✓	✓
	Aircraft warning, radar and navigation system tones	✓	✓
	Synthetic voice message	✓	✓
FSM	Voice communication	✓	✓
	Fuel System	✓	✓
	Electrical System	✓	✓
	Hydraulic System	✓	✓
	Master Caution /Warning system	✓	✓

Table A1. List of ARWA Models and Data [Continued]

Segment	System	RAH-66	AH-64D
VSM	Head tracking prediction algorithms dampening smoothing	✓	✓
	Line of Sight/Ray tracing algorithms	✓	✓
	Databases	✓	✓
	Moving Model Icons	✓	✓
	Intervisibility	✓	✓
TNE	Ownship collision detection	✓	✓
	Dead Reckoning	✓	✓
	AOI screening	✓	✓
	Intervisibility	✓	✓
	Laser Range Finding	✓	✓
	Atmosphere/Magnetic Variation	✓	✓

Table A1. List of ARWA Models and Data [Continued]

## 11. Reuse Sources

The following sources were searched and the results of the searches are given:

### 11.1 Asset Source for Software Engineering Technology (ASSET)

#### 11.1.1 Description

ASSET is a software reuse library and reuse information exchange available to software developers in government, industry, and education. ASSET is sponsored by ARPA's STARS (Software Technology for Adaptable, Reliable Systems) Program to serve as a national resource for the advancement of software reuse across the DoD. The ASSET library, located in Morgantown, WV, is connected to the Internet allowing world-wide access to reusable software assets.

#### 11.1.2 Data Search

A series of pattern searches were performed on the ASSETS catalog document using key words for the ARWA model. The results of these searches is documented below.

Keyword: "navig"

ASSET\_A\_396 Parser Builder Software Bundle

ASSET\_A\_301 Roams Test Report & Lessons Document Learned.

Keyword: "communi"

ASSET\_A\_247 ADA Composer: ADA Design Tool Software Bundle using OOD.

ASSET\_A\_157 ADA Runtime Support for Complex Software Tool Time Critical Embedded Applications.

ASSET\_A\_345 ARPC (Augmented Remote Procedure Software Bundle Call)

ASSET\_A\_517 Cleanroom Engineering Handbook & Document  
Specification Team Practices.

ASSET\_A\_415 Environmental/Tool Integrator User Software  
System Manual.

ASSET\_A\_224 Information Object Modeling Example Software  
Bundle for Air Traffic Control.

\*Some potential for applicability to the ARWA program.

ASSET\_A\_330 Inter-Tool Communications Facility Software  
Bundle (ITCF).

ASSET\_A\_167 Inter-Tool Communications Facility Document  
(ITCF) Final Report.

ASSET\_A\_381 Paradise Document

ASSET\_A\_303 Process Modeling Document

ASSET\_A\_319 Process Notation Development: AAA Document-  
Mag. Notation Article

ASSET\_A\_324 Q an ADA/C/Interprocess Document  
Communications Support Utility

ASSET\_A\_503 Quality Function Deployment Software Bundle

ASSET\_A\_227 Remote Procedure Call Toolkit (RPC) Software  
Tool

ASSET\_A\_175 Requirements Elicitation Process Document

ASSET\_A\_481 RIG Basic Interoperability Data Model  
Document

ASSET\_A\_301 Secure File Transfer Program (SFTP) Document

ASSET\_A\_232 SEE Demonstration Report Software Tool  
Report

ASSET\_A\_323 Software Engineering Courseware, Document  
University of Cincinnati.

Keyword: "flight"

ASSET\_A\_356 ADA/Operating System Interface Software Bundle

Keyword: "controls"

ASSET\_A\_100 GNU SED (Batch Stream Editor) Software Tool

ASSET\_A\_328 UATL (Universal ADA Test Language) Document

Keyword: "weapons"

ASSET\_A\_325 Software Reuse Case Study (Trillium) Document

Keyword: "dynamic"

ASSET\_A\_429 Dynamic Array Package Document

ASSET\_A\_108 Environment/Tool Integrator Software Component

ASSET\_A\_226 Planning and Optimization Tools Software Tool

ASSET\_A\_439 Tailorable ADA Runtime Environment Document  
(TARTE)

ASSET\_A\_234 Terminal Interface Package Software Bundle

Keyword: "physical"

ASSET\_A\_475 ROAMS Testbed Report and Lessons Document  
Learned

Keyword: "simulation"

ASSET\_A\_519 Cleanroom Engineering Handbook: Document  
Organization and Project Formation  
in the Cleanroom.

ASSET\_A\_252 Event Set Manager Package Document

ASSET\_A\_412 External String Management Package Software-  
Component

ASSET\_A\_323 Software Engineering Courseware, Document  
University of Cincinnati.

ASSET\_A\_353 Software Measurement Guidebook Courseware

ASSET\_A\_218 Tasking ADA Simulation Kit (TASKIT) Software  
Bundle

ASSET\_A\_234 Terminal Interface Package, Building  
Software Bundle Blocks

ASSET\_A\_307 Tools/Notation Evaluation Report: Document  
Proto Process Model.

ASSET\_A\_308 Transparent Distributed ADA Runtime Document  
Support

There were no occurrences of the following keywords in the ASSET Catalog:

"sensor"

"aircraft"

"surviv"

"propulsion"

"physical cues"

"cues"  
"cue"  
"queue"  
"fuel system"  
"electrical system"  
"hydraulic"  
"caution"  
"warning"  
"head tracking"  
"line of sight"  
"line-of-sight"  
"moving model"  
"intervisibility"  
"atmosphere"  
"dead reckoning"  
"range finding"  
"collision"

### 11.1.3 Findings

Though many documents and software were found for some keywords, most of the important ARWA keywords led to no information found. Of the documents and software found, many of it is not applicable to the ARWA program. There is little software or documentation that the ARWA program can utilize from ASSETS.

### 11.1.4 Rating

Available relevant software models:	2
Available relevant documentation:	2
Cost, data rights, and electronic access	8
Total score:	4.00

## 11.2 Defense Software Repository System (DSRS)

### 11.2.1 Description

DSRS (formerly RAPID) is an automated library of reusable software development components available to the DoD and other Government agencies, including supporting contractors.

### 11.2.2 Data Search

DSRS was searched for the keyword "mass properties" by the Army Aviation Warfighting Center at Ft. Rucker, AL. The following components were returned:

MAPAC\_Ada\_Transf\_FFT\_Radix8  
MAPAC\_Ada\_Transf\_FFT\_Radix8\_Lookup\_Table  
MAPAC\_Ada\_Transf\_FFT\_Radix2\_Lookup\_Table  
MAPAC\_Ada\_Transf\_Init\_FFT\_Lookup\_Table  
MAPAC\_Ada\_Transf\_Inverse\_FFT\_Radix2  
MAPAC\_Ada\_Transf\_Inverse\_FFT\_Radix2\_Lookup\_Table  
MAPAC\_Ada\_Transf\_Inverse\_FFT\_Radix4  
MAPAC\_Ada\_Transf\_Inverse\_FFT\_Radix4\_Lookup\_Table  
MAPAC\_Ada\_Transf\_Inverse\_FFT\_Radix8  
MAPAC\_Ada\_Transf\_Inverse\_FFT\_Radix8\_Lookup\_Table

MAPAC\_Ada\_Transf\_Scale\_Complex\_By\_Vector\_Length  
 MAPAC\_Ada\_Transf\_Scale\_Complex\_Vect\_To\_Abs\_Amp  
 MAPAC\_Ada\_Transf\_Scale\_Matrix\_By\_Rows\_X\_Columns  
 MAPAC\_Ada\_Transform\_Pac  
 MAPAC\_ada\_Lin\_Gen\_Decompose  
 MARC\_MATRIX\_AUTOMATED\_REDUCTION\_AND\_COUPLING  
 MASPROP\_MASS\_PROPERTIES\_OF\_A\_RIGID\_STRUCTURE  
 MATHEMATICAL\_ROUTINES\_FOR\_ENGINEERS\_AND\_SCIENTISTS  
 MAXIMUM/MINIMUM\_ENVELOPE\_PLOTS  
 MEL21\_Pipe\_Flexibility\_Program\_(CDC Version)  
 MEL21\_Pipe\_Flexibility\_Program\_(IBM Version)  
 MEL21\_Pipe\_Flexibility\_Program\_(Univac Version)

### 11.2.3 Findings

Of the integrated models searched for with the keyword "mass properties", the MASPROP MASS PROPERTIES OF A RIGID STRUCTURE component seems to have the most use for the ARWA program.

### 11.2.4 Rating

Available relevant software models:	2
Available relevant documentation:	2
Cost, data rights, and electronic access	8

Total score: 4.00

## 11.3 Modeling and Simulation Information System (MSIS)

### 11.3.1 Description

The Tactical Warfare and Simulation Technology Information Analysis Center (TWSTIAC) Modeling and Simulation Information Systems (MSIS) is sponsored by the Defense Modeling and Simulation Office (DMSO). The DMSO MSIS is an on-line service available to a large audience of subscribers from government, the military services, academia, and industry, and is designed to serve the Modeling and Simulation (M&S) community by providing current leading edge information on what is happening in the M&S community. The Catalogs of Models and Simulations features information to the subscriber on models and simulations from all the services, the joint staff, and TRANSCOM. The type of data available in this menu includes the Point of Contact (POC), date, description, parameters, uses, and computer requirements data for the several hundred models listed.

### 11.3.2 Data Search

Loral obtained the entire list of models available from MSIS. The sources of these models are War Games, Training Games & Combat Simulation; J-8 M&S Catalog; MOSAIC (Models & Simulations: Army Integrated Catalog); Navy Catalog of Models and Simulations; TRANSCOM System Model Catalog; and US Air Force Rome Laboratory M&S Catalog. Roughly 1000 models exist in these repositories. The AVCATT library was searched for various models by the Army Aviation Warfighting Center at Ft. Rucker, AL. The following components were returned:

ARTOAR	- Attack Helicopter Air-to-Air Fire Control System Simulation Model
HPROBI	- Hit Probability

PS-2	- Propulsion System Performance Simulation
HELIPAC	- Helicopter Piloted Air Combat Model
HAVDEM	- Helicopter Air-to-air Value-Driven Engagement Model
HELSCAM	- Helicopter Scenario Assessment Model
HELMATES II	- Helicopter Launched Missile Antitank Effectiveness Simulation
GPS Map System	- Global Positioning System Map System

### 11.3.3 Findings

The AVCATT search proved very useful in locating not only documentation sources, but also software sources. Many software models from this repository can be utilized in the ARWA device.

### 11.3.4 Rating

Available relevant software models:	7
Available relevant documentation:	7
Cost, data rights, and electronic access	9

Total score: 7.67

## 11.4 Document Cataloging System (DOCATS)

### 11.4.1 Description

The Document Cataloging System (DOCATS) is a data base that identifies all documents in the CCTT library. This data base lists the document name, author name, date, abstract, keywords and other pertinent data that help to identify sources of information. In addition to searches on these fields, searches by weapon system name and use of Boolean operators (and, or, not) are available to narrow or broaden the search. Once a document is identified, a copy can be obtained by identifying the unique document number and title.

### 11.4.2 Data Search

Loral visited Resource Consultants, Inc., the company in charge of the Close Combat Tactical Trainer (CCTT) library. A search was made on AVSCOM AH-64, RAH-66, RWA, Sim Models, Missiles, IR/Laser, and Weapon System Performance. Roughly 5 to 10 documents under each category were found.

### 11.4.3 Findings

Only documentation was available - no software models. DOCATS is not a great source of ARWA information. The POC at RCI is Judith DeNicola, (407)282-151.

### 11.4.4 Rating

Available relevant software models:	0
Available relevant documentation:	3
Cost, data rights, and electronic access	6

Total score: 3.00

## 11.5 Army Reuse Center (ARC)

### 11.5.1 Description

The Army Reuse Center (ARC) is a primary focal point for reuse within the Department of the Army. The ARC was established to support the development and fielding of reliable, high quality systems while reducing the time and resources required to develop and maintain those systems. The mission of the Army Reuse Center is to develop, implement, maintain, and administer a total reuse program that will support the entire software development life-cycle (SDLC). At the heart of the Army Reuse Center is an automated library system that provides user access to a wide range of high quality reusable software components. The library currently contains over 2400 reusable design, code, and document components and represents over 1.8 million lines of code.

### 11.5.2 Data Search

Loral obtained the Army Reuse Center catalog. A non-disclosure agreement needed to be signed in order to obtain any of the information in the Army Reuse Center. Loral desired changes to the non-disclosure agreement to cover legal issues, but the Army Reuse Center explained that a lengthy review would be necessary for this to happen. A non-disclosure agreement was therefore not signed by Loral.

### 11.5.3 Findings

Since a non-disclosure agreement was not signed by Loral, the Army Reuse Center data is unattainable at this time.

### 11.5.4 Rating

Unranked.

## 11.6 Sherikon, Inc.

### 11.6.1 Description

Sherikon, Inc. is under contract under PM CATT to catalog documentation for both RAH-66 and AH-64 aircraft.

### 11.6.2 Data Search

Loral visited the Sherikon office in Orlando, FL, and asked to see the library. The library is still being constructed and no document listing has been produced.

### 11.6.3 Findings

Only documentation was available - no software models. Sherikon could be a source of information once the library becomes operational. The POC at STRICOM is Bob Hale, (407)380-4986.

### 11.6.4 Rating

Available relevant software models:	0
Available relevant documentation:	1

Cost, data rights, and electronic access 5

Total score: 2.00

### 11.7 SPARTA, Inc.

#### 11.7.1 Description

SPARTA performs the V&V for the ARWA project. SPARTA has approved data bases and in-house models which can be used on the ARWA program to validate sensor and weapon modules. These models were approved by AMSAA, Night Vision ESD, and ARL. SPARTA has incorporated these models and data bases into ALWSIM and can exercise that simulation for validation tasks. Standalone versions of some models can also be used for validation.

#### 11.7.2 Data Search

##### SPARTA Validation Models:

###### Sensors:

ACQUIRE Search & Target Acquisition

FLIR 90 FLIR performance

IMAGE INT. Image Intensifier Performance/TV Performance

PHI Laser Target Acquisition

TARGET CONTRAST Optical Contrast

###### Weapons:

INDIRECT FIRE EFFECTS HE/ICM P<sub>k</sub>

INCURSION AD Effects - Guns & Missiles

GAMES Smart Munition Effects

LELAWS Laser Weapon Effects

DMEWS HPM Weapon Effects

###### Environment:

EOSAEL Natural Atmosphere, Smoke, Dust

##### SPARTA Validation Data Bases:

###### Weapons:

DIRECT FIRE Accuracy (Bias, Dispersion)

DIRECT FIRE Vulnerability, P<sub>k</sub>/HIT

DIRECT FIRE Timeliness

DIRECT FIRE Vehicle Characteristics

INDIRECT FIRE Delivery Accuracy (MPI, Precision)

INDIRECT FIRE Lethal Area

###### Scenarios:

HRS1, HRS29, HRS14

#### 11.7.3 Findings

Only models for performing V&V are available.

#### 11.7.4 Rating

Available relevant software models: 8

Available relevant documentation: 5  
Cost, data rights, and electronic access 5

Total score: 6.00

### **11.8 Public Ada Library (PAL) (Ada Software Repository)**

#### **11.8.1 Description**

The Public Ada Library (PAL) is a collection of Ada programs, tools, and educational materials. Source code can be retrieved over the Internet via FTP (wuarchive.wustl.edu).

#### **11.8.2 Data Search**

Loral obtained the PAL catalog of reusable software from wuarchive.wustl.edu. The listing of software components included screen routines, math libraries, and simple algorithms. The listing of software development tools included many Ada analysis tools.

#### **11.8.3 Findings**

The basic software components, though not ARWA specific models, could be used in some applications for the ARWA program. Software development tools could also be used to some extent.

#### **11.8.4 Rating**

Available relevant software models: 2  
Available relevant documentation: 3  
Cost, data rights, and electronic access 8

Total score: 4.33

### **11.9 Ada Joint Program Office (AJPO) and AdaIC**

#### **11.9.1 Description**

Source code from some AJPO-sponsored projects is available through the Ada Information Clearinghouse and the AJPO host (ajpo.sei.cmu.edu) on the Internet. Source code may be retrieved via FTP.

#### **11.9.2 Data Search**

A listing of available documentation and software was retrieved from the AJPO Internet address.

#### **11.9.3 Findings**

Very limited software is available. The documentation centered around Ada standards.

#### **11.9.4 Rating**

Available relevant software models: 1  
Available relevant documentation: 1  
Cost, data rights, and electronic access 7

Total score: 3.00

**11.10 National Technical Information Services (NTIS)****11.10.1 Description**

NTIS is a self-supporting publishing agency for the U.S. Department of Commerce. It provides a free catalog of the software available from the Federal Computer Products Center, which is a clearinghouse for over 3500 products from about 100 Federal agencies.

**11.10.2 Data Search**

Loral obtained the NTIS catalog of software.

**11.10.3 Findings**

Software has limited rights and has cost involved. An ARWA software search turned up the following potentially reusable software models:

Communications model:	Terrain-Integrated Rough-Earth Model (TIREM), \$140, Point-to-point radio transmission loss
Navigation model:	Mapping Datum Transformation Software (MADTRAN), \$55, Coordinate conversion program

No ARWA documentation was available.

**11.10.4 Rating**

Available relevant software models:	2
Available relevant documentation:	0
Cost, data rights, and electronic access	6
Total score:	2.67

**11.11 AdaNET****11.11.1 Description**

AdaNet is a component of the Repository Based Software Engineering (RBSE) program sponsored by NASA. RBSE is a research and development program designed to effectively transfer software engineering technology among U.S. government, industry, and academia. The purpose of RBSE is to support the adoption of software reuse through repository-based software engineering. The program provides a repository that: facilitates the selection, acquisition, integration, and reuse of software components; and promotes common software engineering practices and standards. The AdaNET Repository currently contains reusable, public domain software from the following sources:

- Ada Software Repository (Army/ASR)
- Jet Propulsion Lab (NASA/JPL)
- DoD/STARS
- Educational Institutions.

The following collections are available on AdaNet.

**AdaNet Collections:**

1. AI/Expert Systems.SF
2. ASV3 Support.SG

3. Education.SH
4. Human Rated Systems.SI
5. Image Processing and Analysis.SJ
6. Information Management.SK
7. Language Features and Constructs.SL
8. Legal Issues.SM
9. Library Interfaces and Protocols.SN
10. Lifecycle Methods and Tools.SO
11. Metrics.SP
12. Routines and Algorithms.SQ
13. Standards.SR
14. System Support.SS
15. User Interfaces.ST
16. Samples.SU

#### 11.11.2 Data Search

Loral is a member of AdaNET. The above collections were searched, and there were no models directly applicable to the ARWA project. There are some generic math algorithms and some metrics available which may be somewhat useful.

#### 11.11.3 Findings

Not very many models are available for the ARWA project.

#### 11.11.4 Rating

Available relevant software models:	1
Available relevant documentation:	0
Cost, data rights, and electronic access	8
Total score:	3.00

### 12. Conclusions

The final ratings are ordered as follows:

7.67	Modeling and Simulation Information System (MSIS)
6.00	Sparta, Inc.
4.33	Public Ada Library (PAL) (Ada Software Repository)
4.00	Defense Software Repository System (DSRS)
4.00	ASSET Source for Software Engineering Technology
3.00	Ada Joint Program Office (AJPO) and AdaIC
3.00	Document Cataloging System (DOCATS)
3.00	AdaNET
2.67	National Technical Information Services (NTIS)
2.00	Sherikon, Inc.
Unranked	Army Reuse Center (ARC)

These rankings reflect the level of reusability of existing data for the ARWA program.

### 13. Bibliography

- [ReNews 93] ReNews (c) - The Electronic Software Reuse and Re-engineering Newsletter. Vol. 3 No. 2 - October 1993

[MSIS]            The Modeling and Simulation Information System brochure,  
                  Institute of Simulation and Training.

[CATT]            CATT Data Base Support Libraries brochure, STRICOM

[ARC]             Army Reuse Center brochure, Army Reuse Center

## **APPENDIX B**

### **REUSE DESIGN AND CODING GUIDELINES**

#### **20. Introduction**

This set of reuse guidelines for designs and code is based on published industry and in-house reports and documentation. Most of the guidelines are generic and non-language specific, except where noted. The sequence of guidelines does not imply rank or importance. This listing is an overview only. Detailed definitions, descriptions, and examples are provided in the references associated with each guideline. Finally, the purpose of this list is to provide a standard set of guidelines to be used within Loral and by its subcontractors for the ARWA project.

#### **21. Design Guidelines**

##### **1. Component Structure [Lea 93]**

- a. Identify and encapsulate commonalty and variability.
- b. Separate interfaces and implementations.
- c. Identify and isolate context and policy from functionality.
- d. Link documentation to code.
- e. Link tests to code.
- f. Use tools when target languages do not support sufficient interface, composition, and/or parameterization constructs.

##### **2. Interfaces [Lea 93]**

- a. Minimize the number of names per name space (scope).
- b. Minimize implementation-dependence of interfaces.
- c. Refine interfaces by extending and adding properties.
- d. Optimize components via specialization.

##### **3. Composition [Lea 93]**

- a. Identify and minimize import requirements.
- b. Identify and minimize interference among helpers.
- c. Use layering to define complex components using simple ones.
- d. Implement policy on top of mechanism.

##### **4. Parameterization [Lea 93]**

- a. Use parameterization to abstract away contextual variability.
- b. Use instantiation to generate components.

##### **5. Isolate the hardware, software, and database management system implementation functions. [Hooten 89]**

This allows minimum impact when enhancing or correcting the system.

6. Extend the design to encompass the entire set of end users, i.e., software developers, maintainers, and reusers. [Hooten 89]

7. Isolate all hardware and operating system dependencies. [Hooten 89]

These types of "calls" should be packaged in small software interface routines that can be tailored to the environment or replaced with equivalent modules in a subsequent environment.

8. Isolate items which are likely to change. [Hooten 89]

9. Don't plan to reuse software components that have to be modified more than 30 percent, but extract design and algorithmic details instead. [Hooten 89]

10. Keep interfaces as simple and application-nonspecific as possible

11. Think in higher levels of abstractions for functions, data, and processes.

[Alexandris 86]

*Function abstractions* (e.g., subprogram interface specifications) are designs based on the user only being aware of the input-output specification while the implementation is hidden from the user. The same function may be reused for a variety of data.

*Data abstractions* (e.g., Ada packages) are designs in which the data and several function implementations are hidden from the user, possibly with superimposed hierarchical inheritance on data abstractions, facilitating dynamic determination of the function to be invoked. Data objects may be reused for various operations that may be applied to them.

*Process abstractions* (e.g., Ada tasks) operate like data abstractions, only they have an independently executing thread of control that determines the order in which operations become available for execution and include concurrent processes that may communicate through shared data in global memory and distributed processes that communicate by message passing.

12. Design more for flexibility, not generality. [Parnas et. al. 89]

Allow for easy modifications within the domain that are reasonable to occur in the future, not every possibility. This would make the code too cumbersome and slow.

## 22. Coding Guidelines

### 22.1 General

1. Keep modules small and simple, i.e., minimize the number of functions per module. [Hooten 89]

2. Each module should contain clear documentation regarding its purpose, capabilities, constraints, interfaces, and required resources. [Hooten 89]

3. Whenever possible, use a portable, high-order programming language. [Hooten 89]
4. Avoid compiler-specific instructions. [Hooten 89]
5. Adhere to common coding standards, conventions, and styles. [Hooten 89]
6. Don't assume that a given feature is present or not present in the system. [Parnas 72]
7. Avoid chains of data transforming components. [Parnas 72]

A chain of data transforming components is a sequence of components, each receiving data from the previous component and then processing the data into another format for the next component. When the chain is broken, the inputs become incompatible.

8. Minimize the "uses" structure. [Parnas 72]

One may end up with a system in which nothing works until everything works. For example, while it may seem wise to have an operating system scheduler use the file system to store its data rather than use its own disk routines, the result will be that the file system must be present and working before any task scheduling is possible.

9. Clearly document all error conditions. [Hooten 89]
10. Isolate machine-dependent operations. [Hooten 89]
11. Isolate operating system-dependent operations. [Hooten 89]
12. Isolate database management system-dependent operations. [Hooten 89]
13. Use non-exotic algorithms, whenever possible. Otherwise, be sure to fully document the algorithm in the specification or some other visible place in the code. [Hooten 89]
14. Avoid table size constraints. [Hooten 89]

## 22.2 Ada Language

1. The specification (portion of code that defines and initializes the program variables) must be readable and understandable. It must be well documented so as to fully describe each parameter that it uses and its interfaces to other packages. [Hooten 89]
2. Use information hiding techniques. System details that are likely to change independently should be hidden in the bodies and assumptions unlikely to change should be placed in the specification. [Parnas et. al. 89]

For example, every data structure is private to one module; it may be directly accessed by one or more programs within the module but not by programs outside the module. Any other program that requires information stored in a module's data structures must obtain it by calling programs on the module interface.

## 23. Bibliography

- [Alexandris 86] Alexandris, N. February 1986. "Adaptable Software and Hardware Problems and Solutions." *Computer*. Vol. 18. No. 2. pp. 29-39.
- [Hooten 89] Hooten, M. 1989. *Software Reuse Methodology and Checklists*. FACC-TR-1113. Ford Aerospace/Space Information Systems Division. (Now Loral Space Information Systems.) Houston, Texas. Company Proprietary.
- [Lea 93] Lea, D. November 1993. *WISR'93 Design-for-Reuse Working Group Report*. Workshop on Institutionalizing Software Reuse held on November 1 - 4, 1993. IBM. Owego, New York.
- [Ogush 93] Ogush, M. 1993. "C Design and Coding Guidelines for Reuse." Hewlett-Packard. Palo Alto, California.
- [Parnas et. al. 89] Parnas, D., P. Clements, and D. Weiss. 1989. "Enhancing Reusability with Information Hiding." *Software Reusability Vol. I - Concepts and Models*. Association for Computing Machinery Press. pp. 141-157.
- [Parnas 72] Parnas, D. December 1972. "On the Criteria to be Used in Decomposing Systems into Modules." *Communications of the ACM*. Vol. 15.